MONOTONIC STACK

BRUCE NAN

STACK REVIEW

- Stacks are linear data structure.
- All deletions and insertions occur at one end of the stack known as the TOP.
- Data going into the stack first, leaves out last.
- Stacks are also known as LIFO data structures (**L**ast-**I**n, **F**irst-**O**ut).

BASIC STACK OPERATIONS

- push – Adds an item to the top of a stack.
- pop – Removes an item from the top of the stack and returns it to the user.
- stack top (top, peek) – Copies the top item of the stack and returns it to the user; the item is not removed, hence the stack is not altered.

MONOTONIC STACK

- **Monotonic stack** is actually a stack.

- It just uses some greedy logic to keep the elements in the stack orderly (monotone increasing or monotone decreasing) after each new element putting into the stack.

EXAMPLE 1: NEXT GREATER NUMBER

PROBLEM DESCRIPTION

- Given an array with n number, return an array of equal length. The corresponding index stores the next larger element, if there is no larger element, store -1.

- Sample input:

  [2, 1, 2, 4, 3]

- Sample output:

  [4, 2, 4, -1, -1]

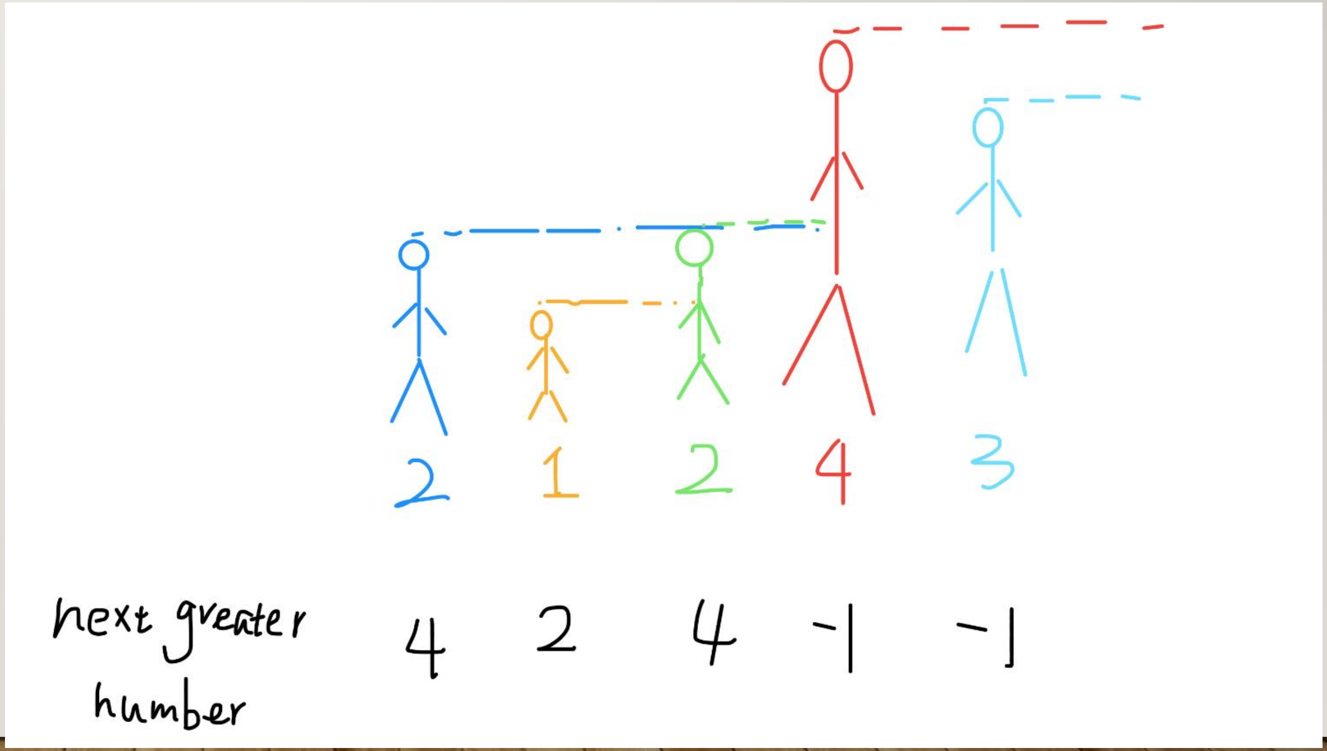
BRUTE FORCE

- For each number, iterate through the array to find out the next greater number.
- Time complexity O(n^2)

- Do we really need to loop through the array?

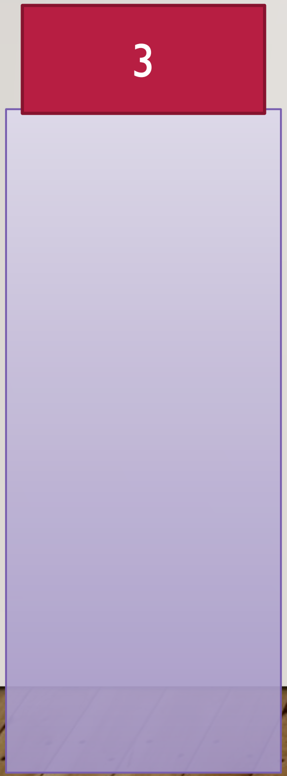
MONOTONIC STACK

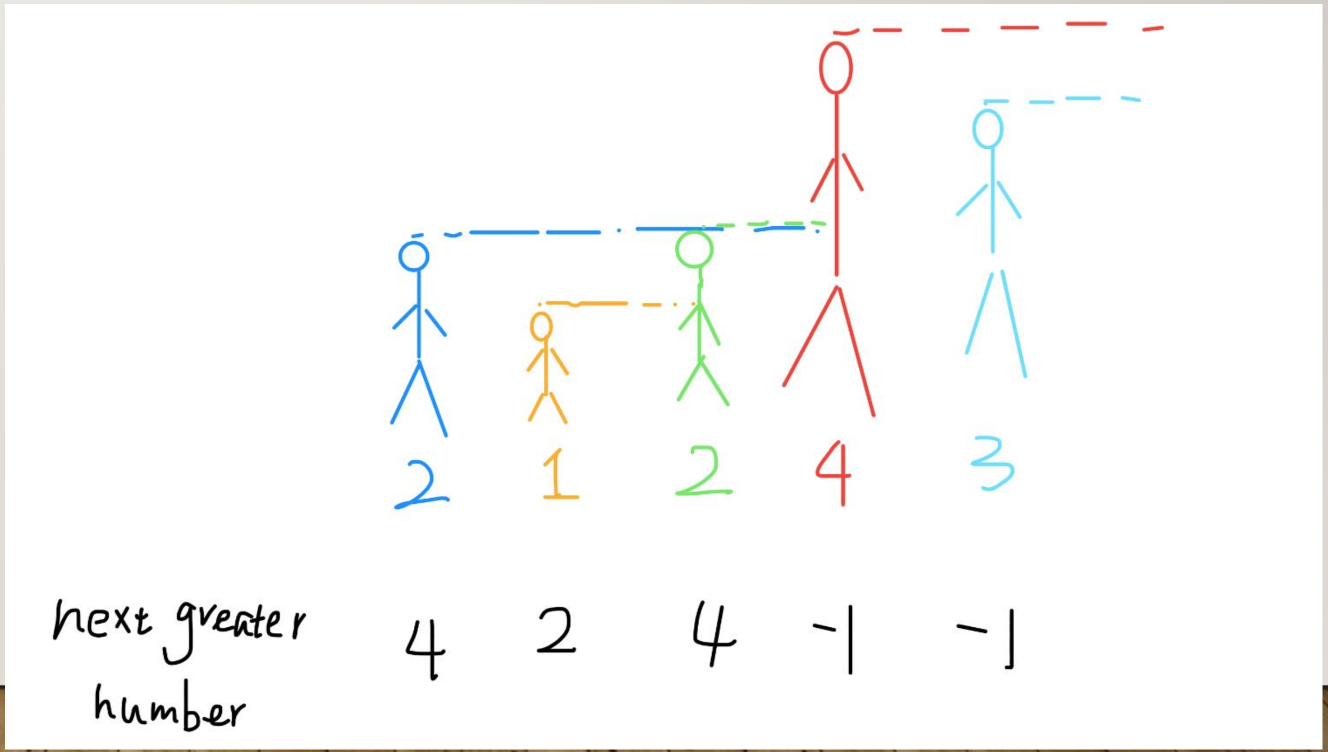
- Idea:
  - Treat elements in the array as people standing side by side
  - To find the next greater number is just to see which person is the first one on your right who blocks other persons
  - Iterate from right to left and use a monotonic stack to keep the higher people on your right



MONOTONIC STACK

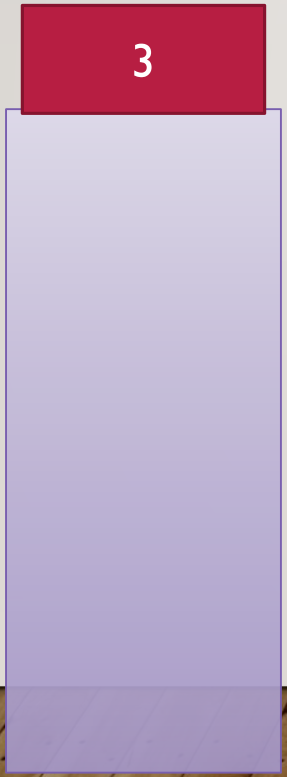
index	1	2	3	4	5
arr	2	1	2	4	3
ans					

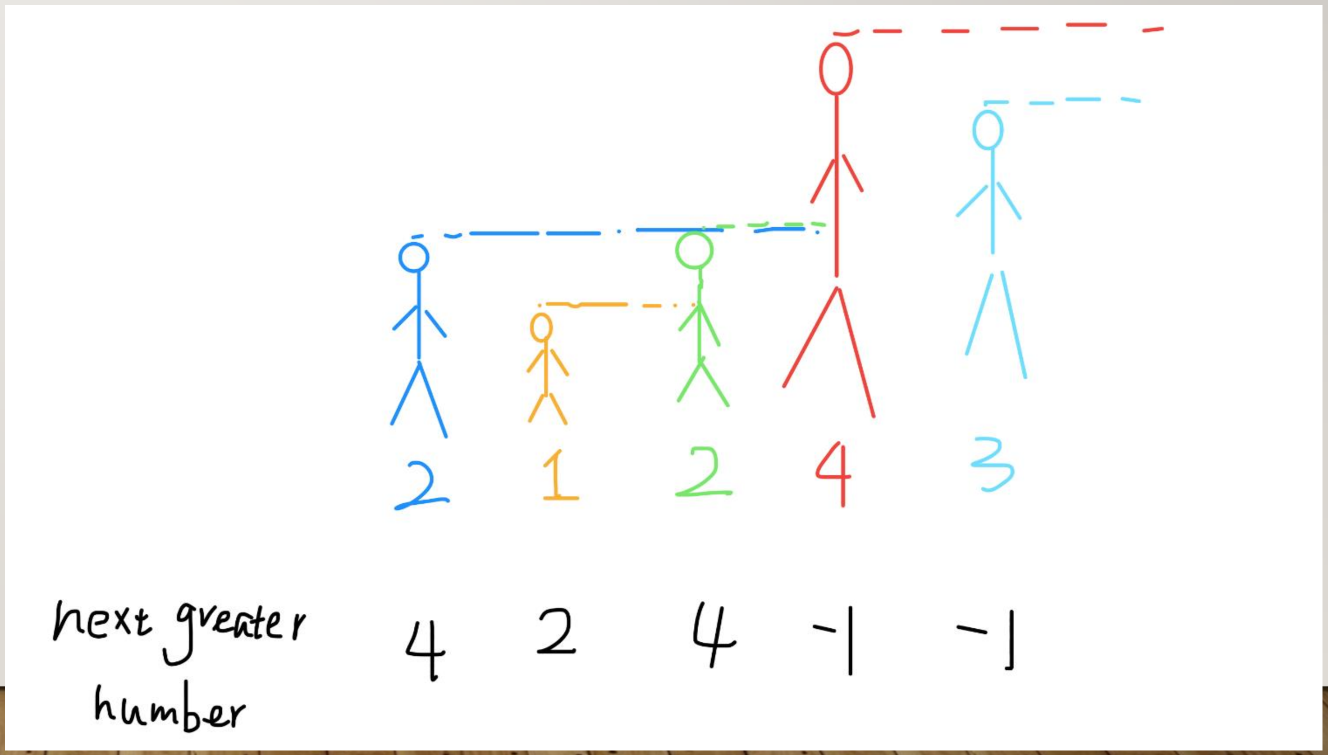




MONOTONIC STACK

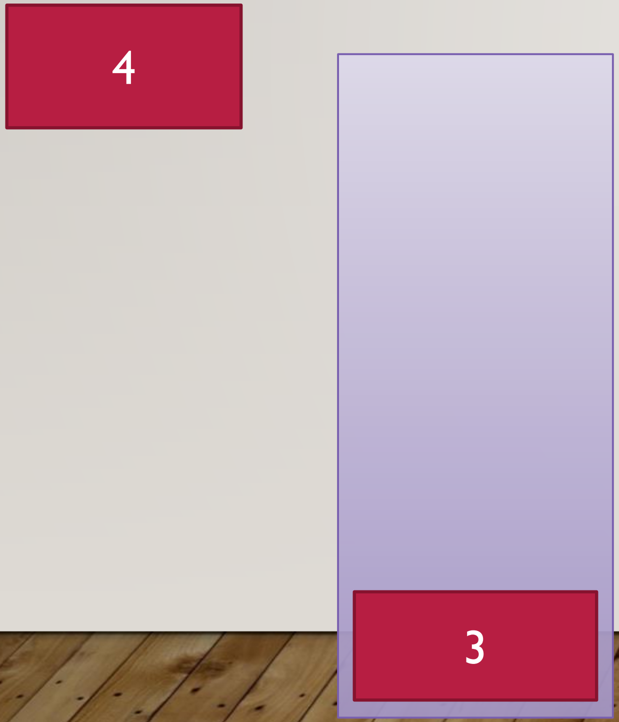
index	1	2	3	4	5
arr	2	1	2	4	3
ans					-1

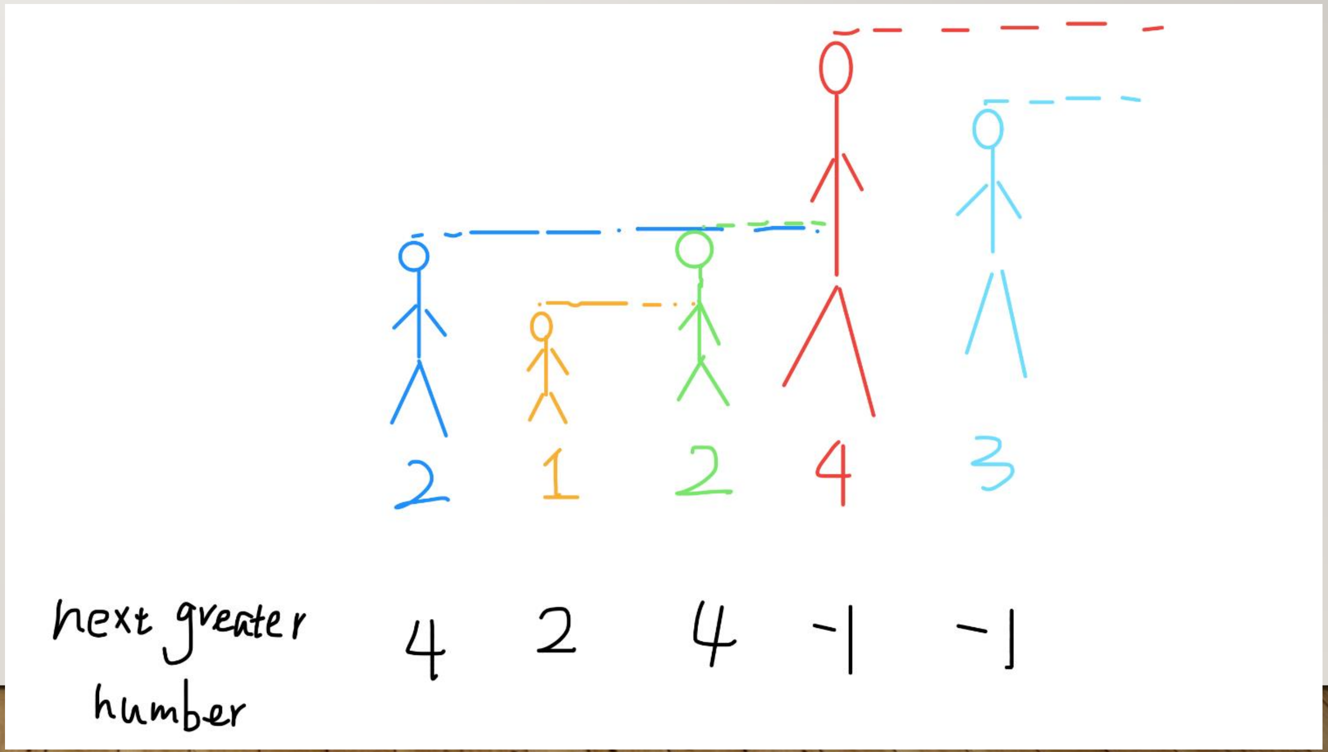




MONOTONIC STACK

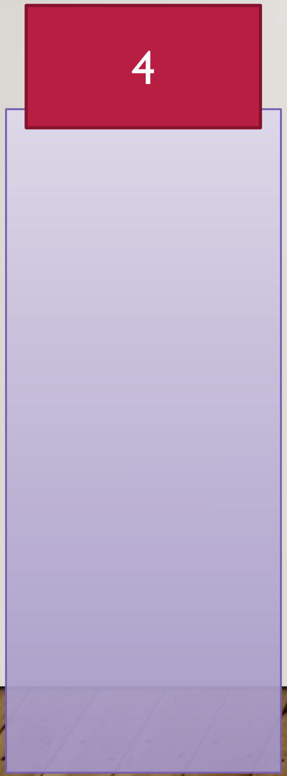
index	1	2	3	4	5
arr	2	1	2	4	3
ans					-1

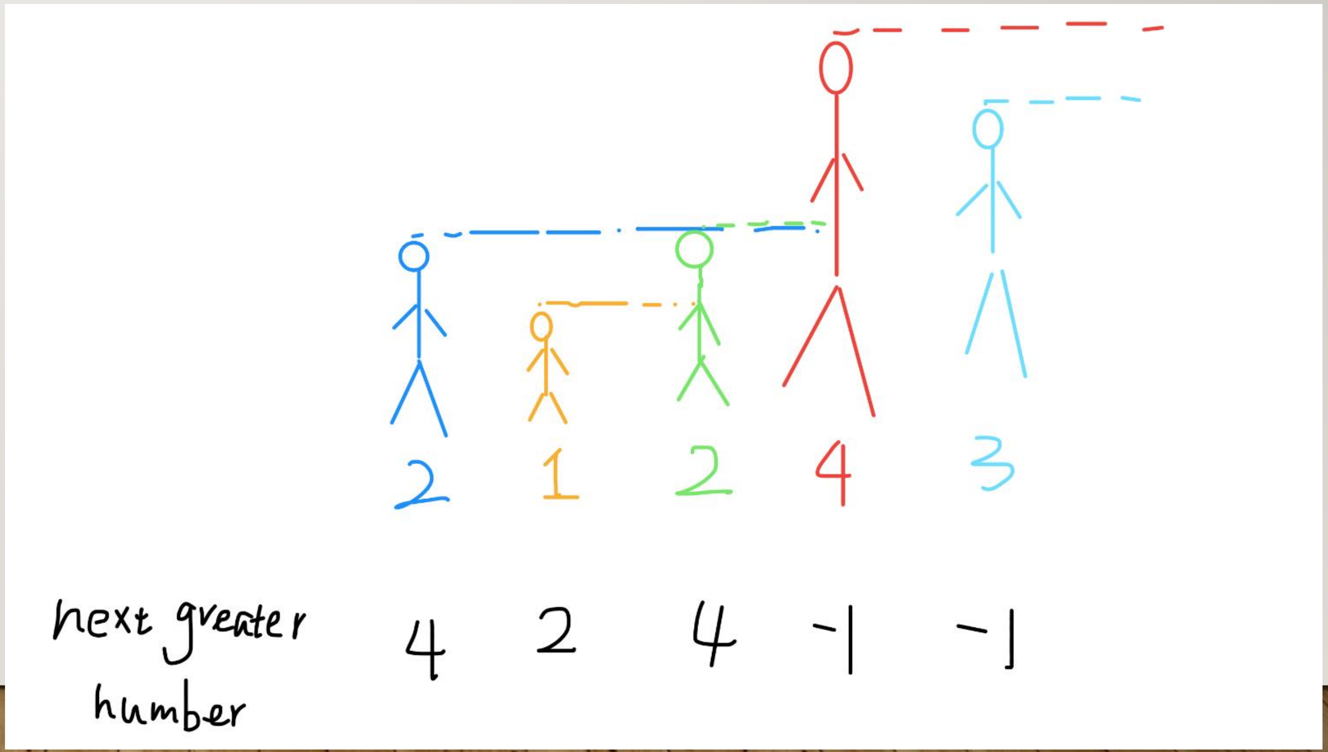




MONOTONIC STACK

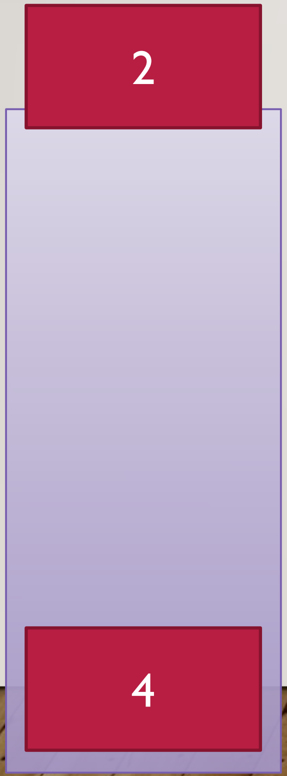
index	1	2	3	4	5
arr	2	1	2	4	3
ans				-1	-1

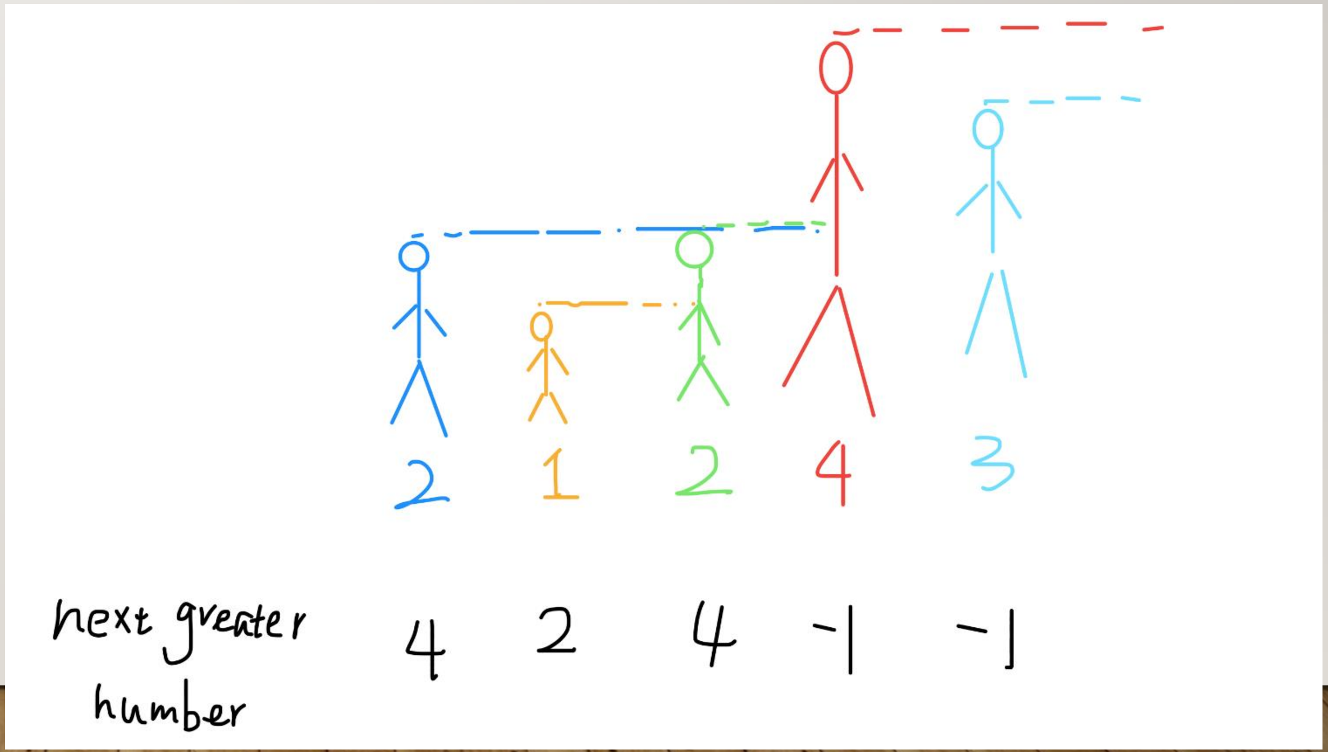




MONOTONIC STACK

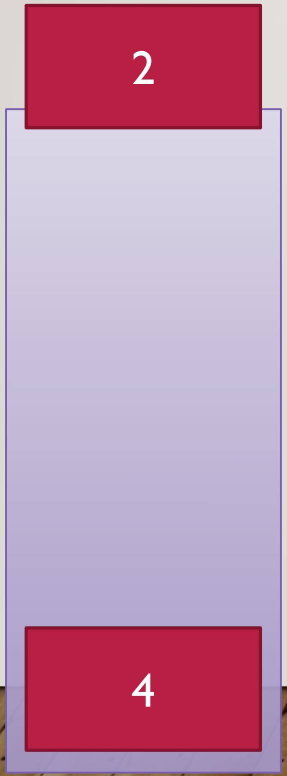
index	1	2	3	4	5
arr	2	1	2	4	3
ans				-1	-1

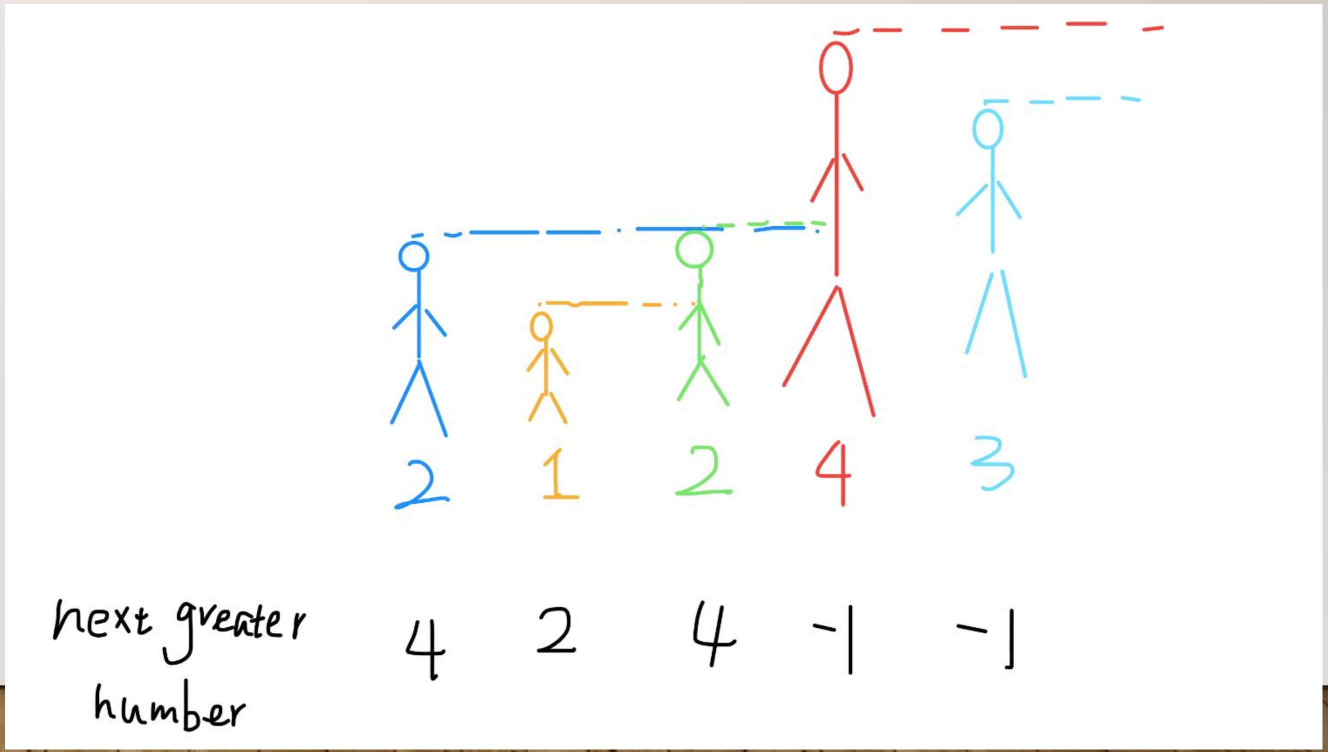




MONOTONIC STACK

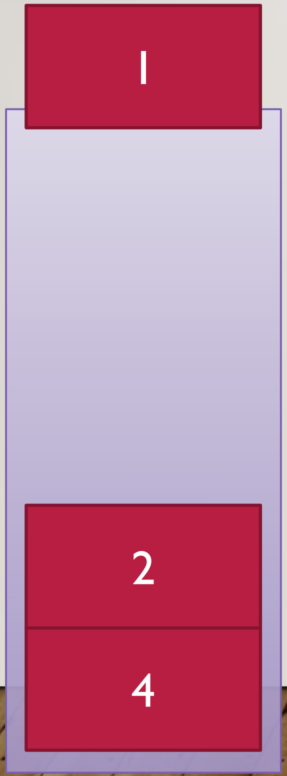
index	1	2	3	4	5
arr	2	1	2	4	3
ans			4	-1	-1

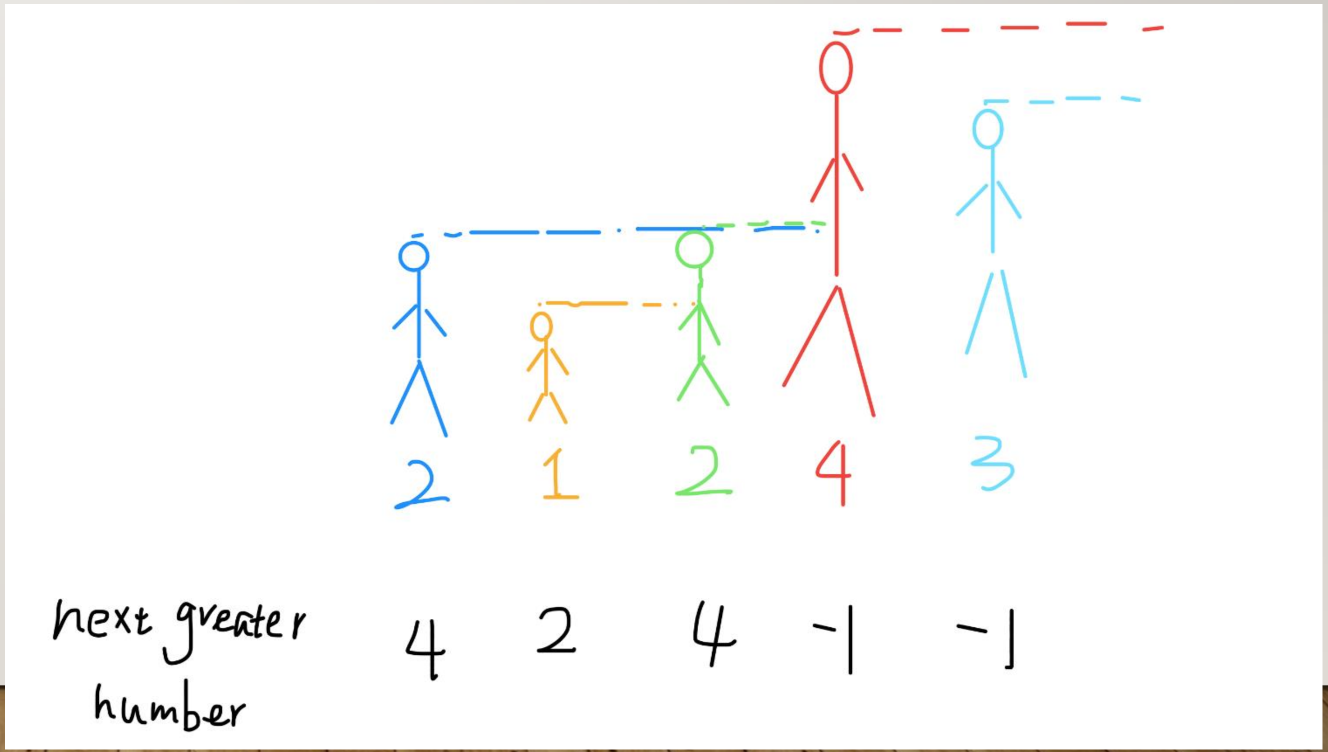




MONOTONIC STACK

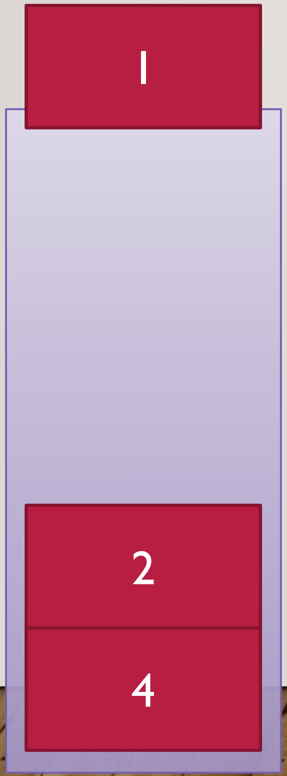
index	1	2	3	4	5
arr	2	1	2	4	3
ans			4	-1	-1

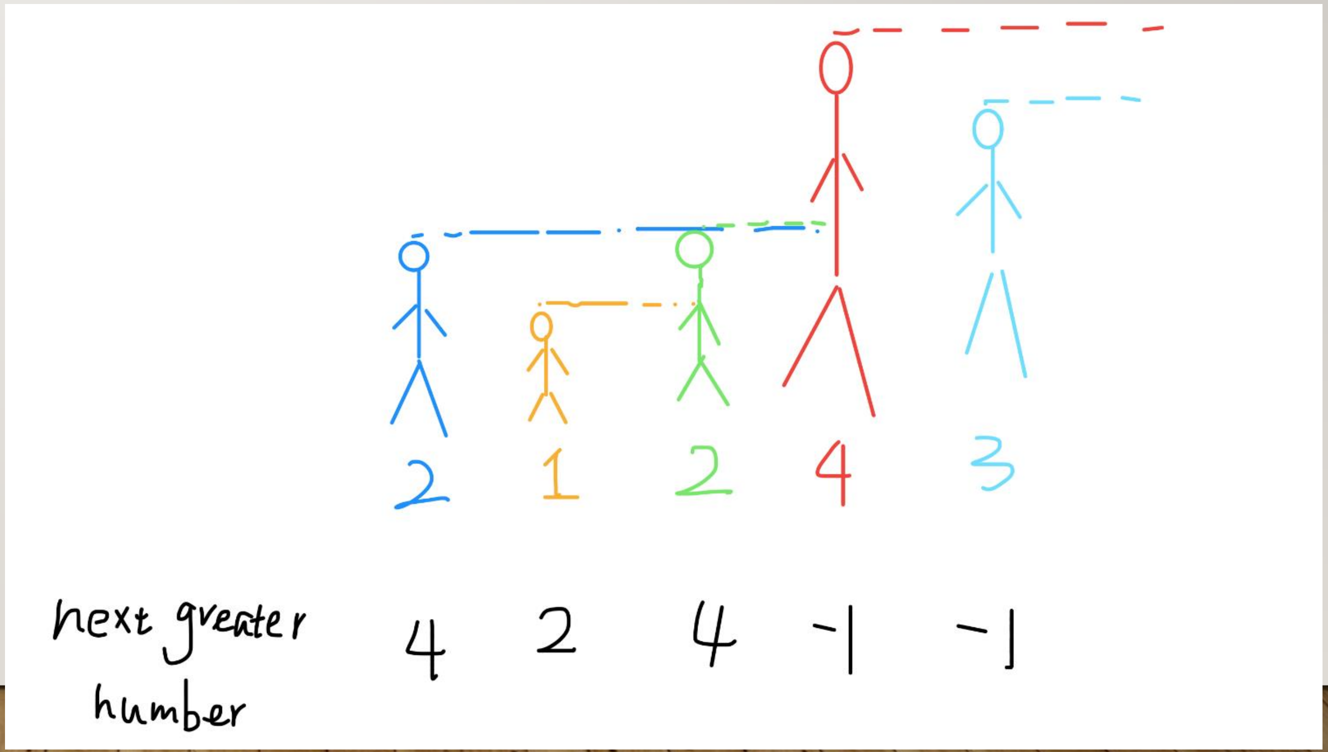




MONOTONIC STACK

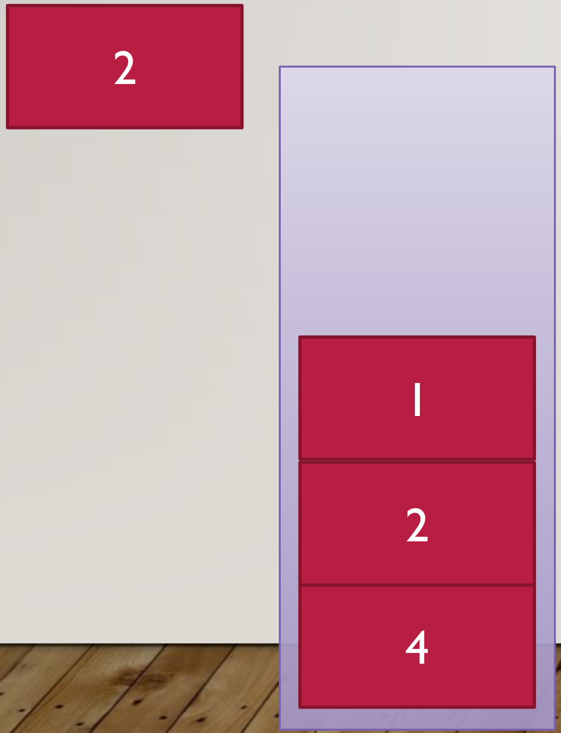
index	1	2	3	4	5
arr	2	1	2	4	3
ans		2	4	-1	-1

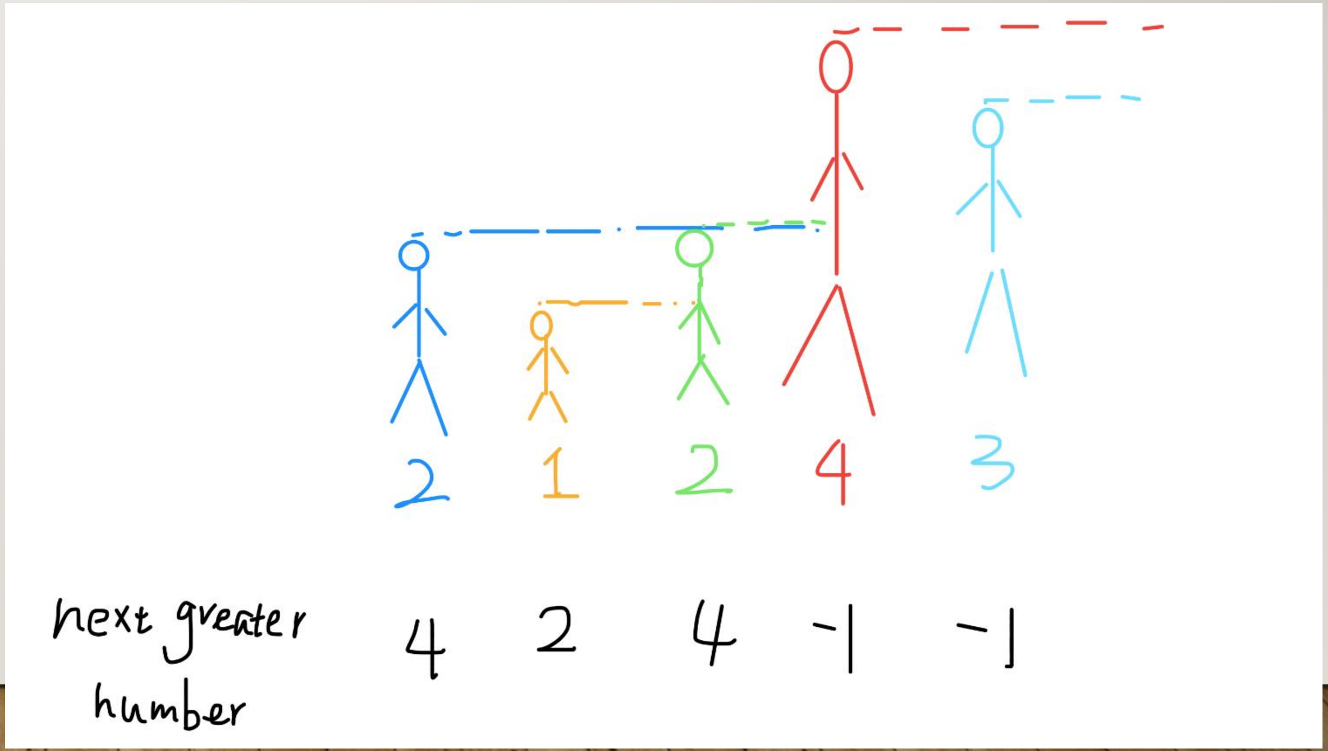




MONOTONIC STACK

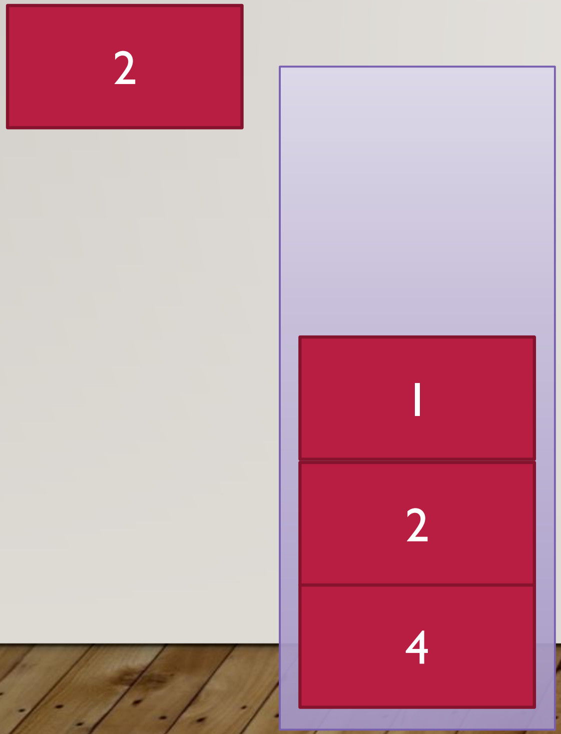
index	1	2	3	4	5
arr	2	1	2	4	3
ans		2	4	-1	-1

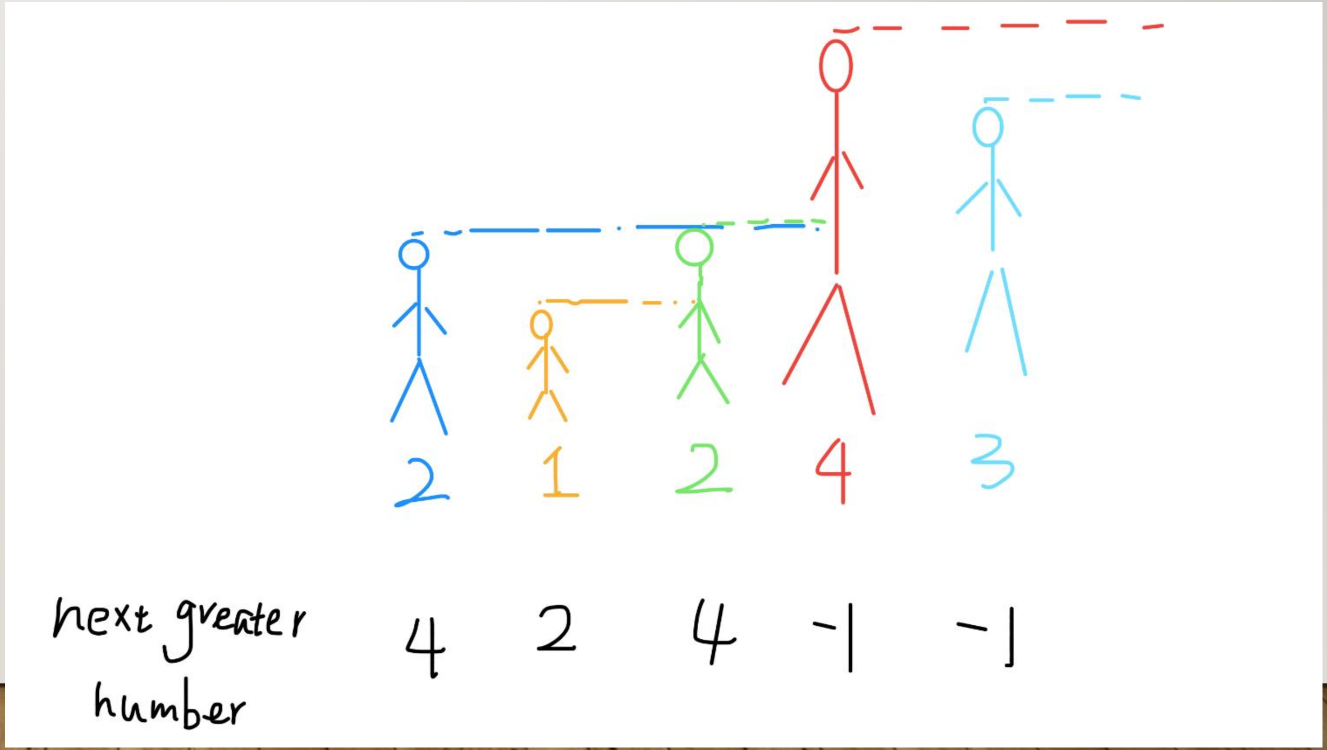




MONOTONIC STACK

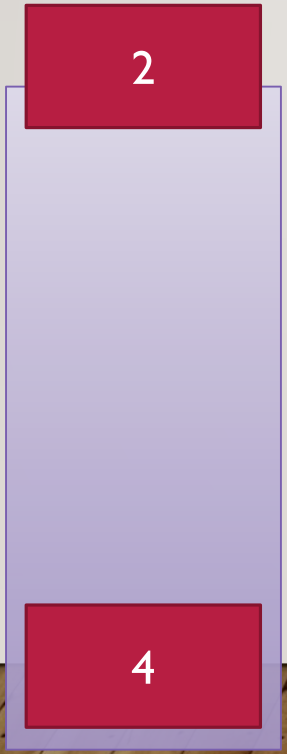
index	1	2	3	4	5
arr	2	1	2	4	3
ans		2	4	-1	-1

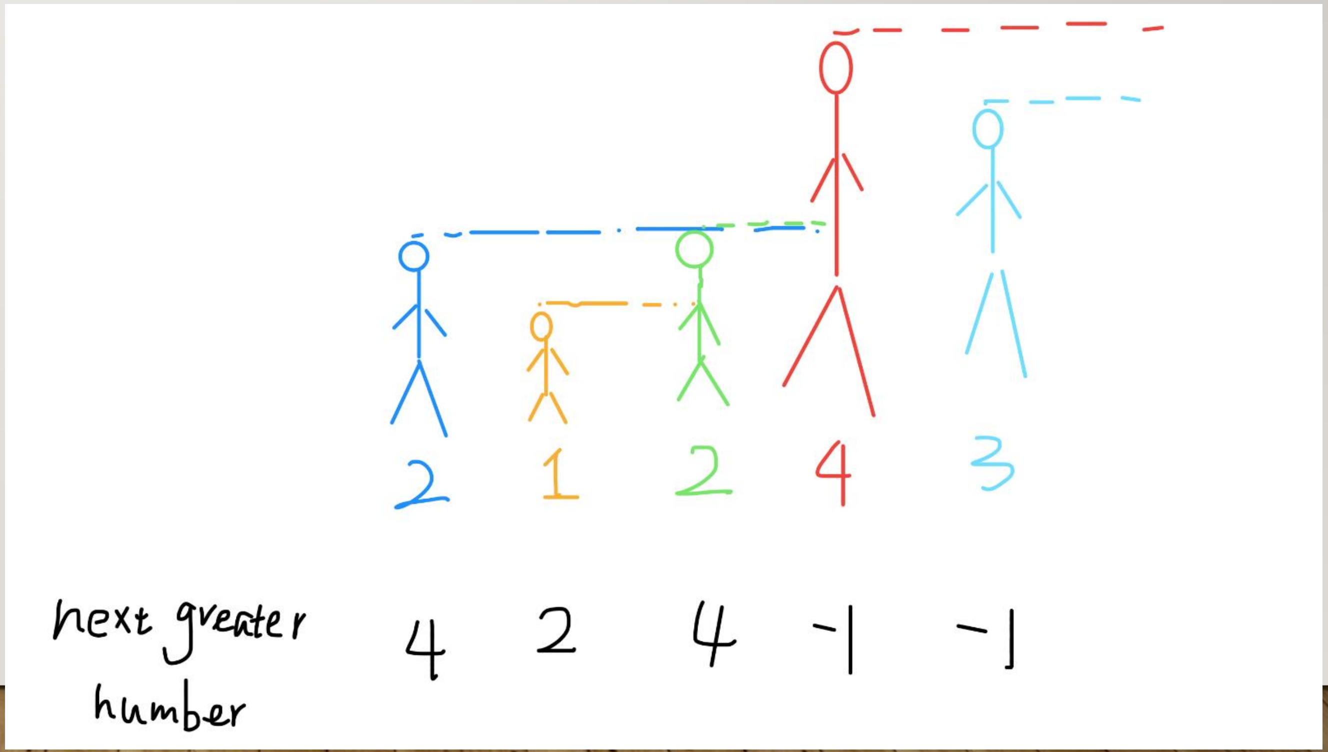




MONOTONIC STACK

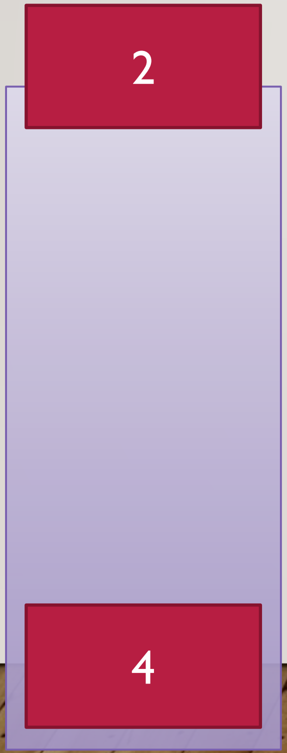
index	1	2	3	4	5
arr	2	1	2	4	3
ans	4	2	4	-1	-1

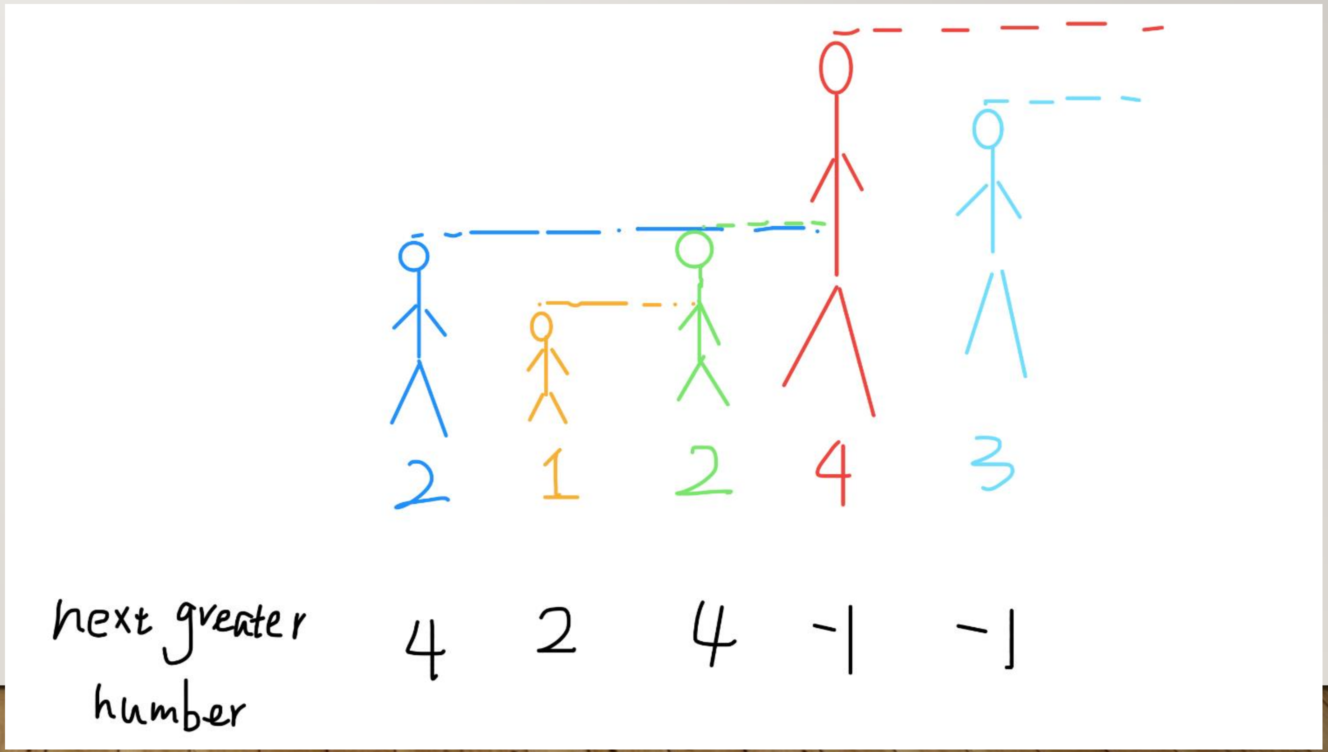




MONOTONIC STACK

index	1	2	3	4	5
arr	2	1	2	4	3
ans	4	2	4	-1	-1





PSEUDOCODE

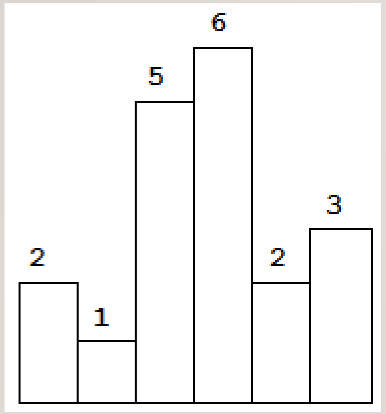
```
for (int i = n-1; i >= 0; i--){
  while( !stack.isEmpty() && stack.peek() <= arr[i] ) {
    stack.pop();
  }
  ans[i] = stack.isEmpty()? -1:stack.peek();
  stack.push(arr[i]);
}
```

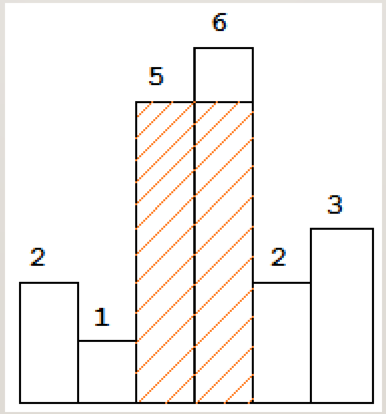
- Each number is pushed into stack once and popped from stack once
- Time complexity O(n)
- Space complexity O(n)

EXAMPLE 2 LARGEST RECTANGLE IN HISTOGRAM

PROBLEM DESCRIPTION

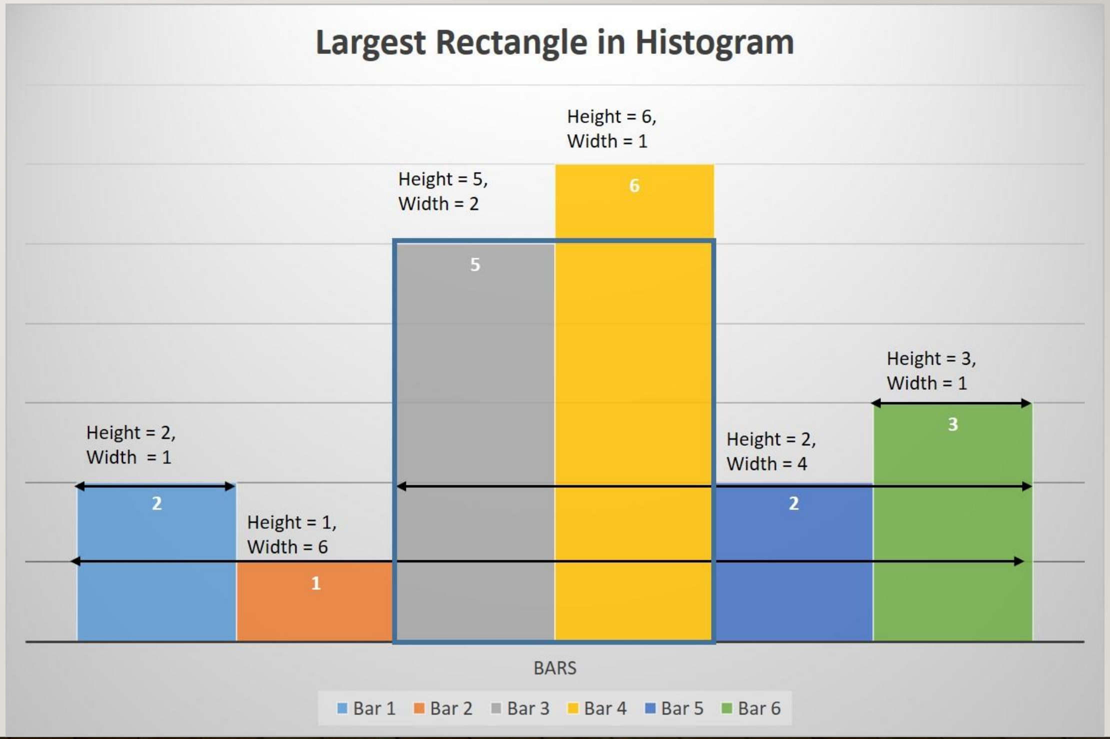
- Leetcode: Largest Rectangle in Historgram
- Given *n* non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of largest rectangle in the histogram.
- Input: [2, 1, 5, 6, 2, 3]   Output: 10





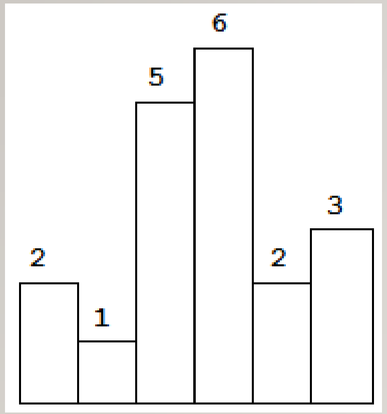
ANALYSIS

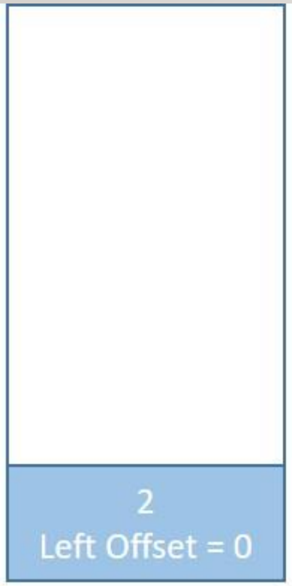
- For each histogram bin i,
  - Find out which bin j is the most closest bin to i on i's left and height[j] < height[i]
  - Find out which bin k is the most closest bin to i on i's right and height[k] < height[i]
  - The largest rectangle with height[i] has the width of $k - j - 1$
  - Thus, the rectangle area is $(k-j-1)*\text{height}[i]$



MONOTONIC STACK

- Idea:
    - Push array elements into a stack as long as the present top of the stack is shorter than the element being pushed.
    - If the stack top is larger than the present element, then the present element provides the right hand side offset for the width, calculate the area of the rectangle with top element as the height.
    - Note that the left offset can be identified by checking the stack element right below the top.
    - After calculating the area, pop the stack.
    - Repeat as long as the top of the stack is larger than the present element.



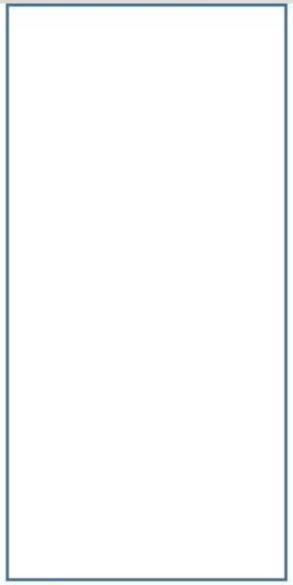


PUSH

Stack Top = 2

Largest Rectangle = 0

{1, 5, 6, 2, 3}



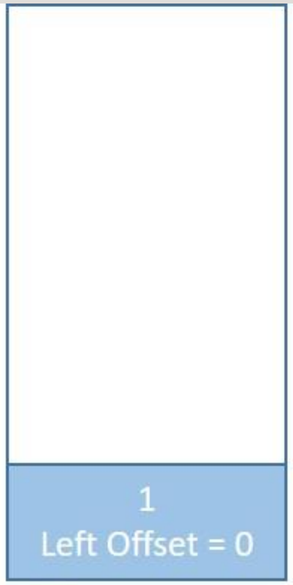
POP

Stack Top = EMPTY

Left offset = 0
Right offset = 1
Rectangle =
[2 x [1 - 0]] = 2

Largest Rectangle = 2

{1, 5, 6, 2, 3}

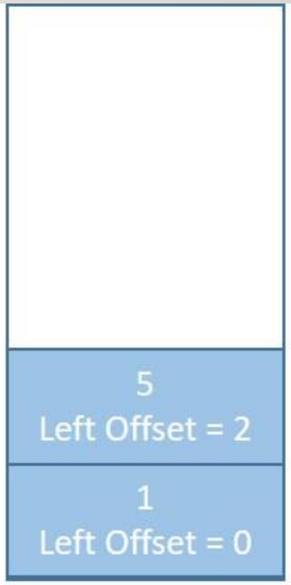


PUSH

Stack Top = 1

Largest Rectangle = 2

{5, 6, 2, 3}

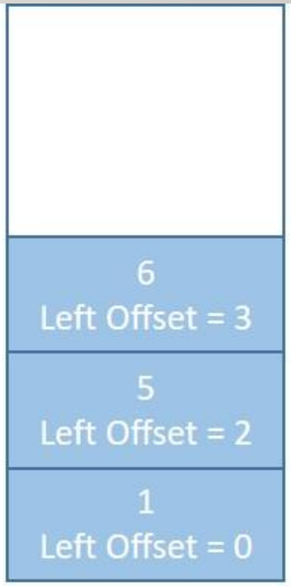


PUSH

Stack Top = 5

Largest Rectangle = 2

{6, 2, 3}

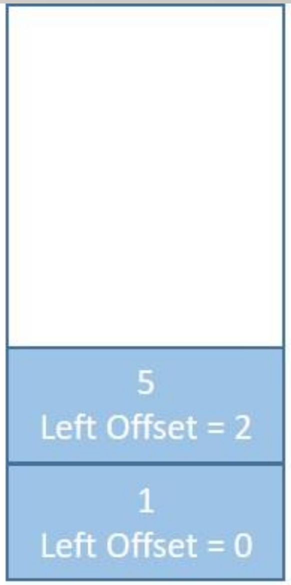


PUSH

Stack Top = 6

Largest Rectangle = 2

{2, 3}



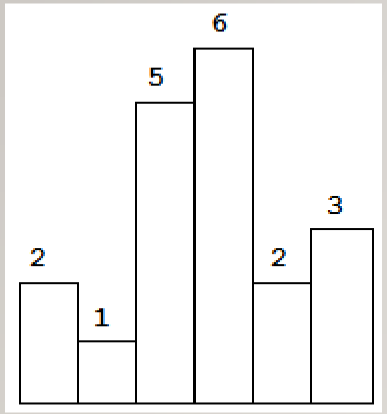
POP

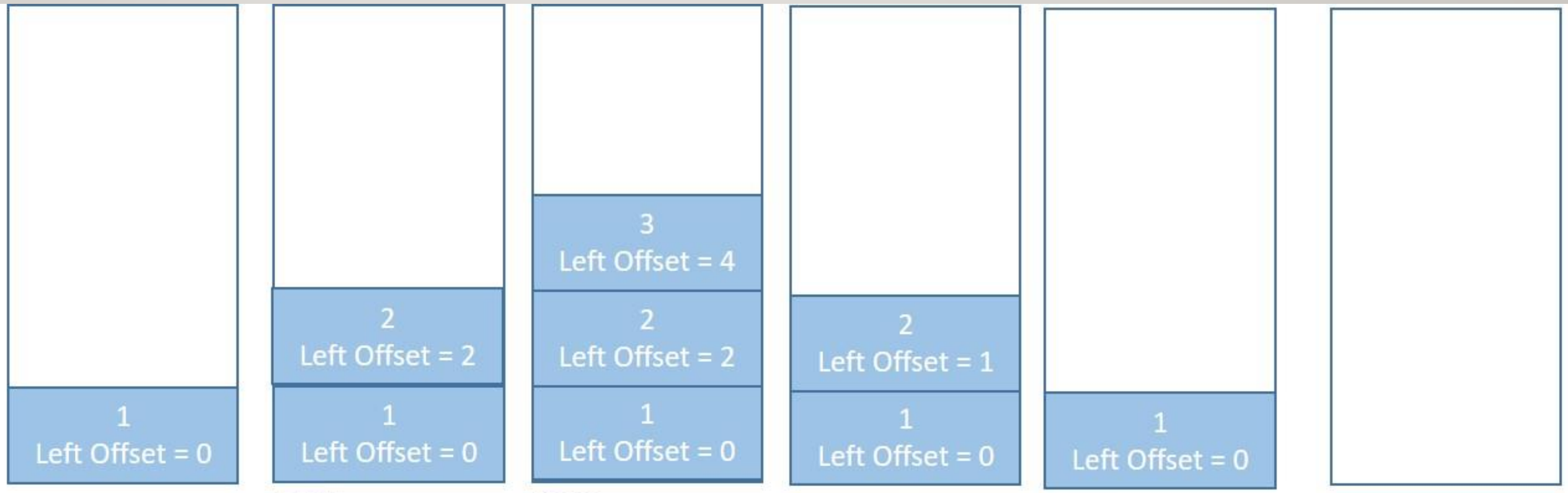
Left Offset = 3
Right Offset = 4
Rectangle =
[6 x (4 – 3)] = 6

Stack Top = 5

Largest Rectangle = 6

{2, 3}





POP

Left Offset = 2
Right Offset = 4
Rectangle =
[5 x (4 – 2)] = 10

Stack Top = 1

Largest Rectangle = 10

{2, 3}

PUSH

Stack Top = 2

Largest Rectangle = 10

{3}

PUSH

Stack Top = 3

Largest Rectangle = 10

{}

POP

Left Offset = 4
Right Offset = 5
Rectangle =
[3 x (5 – 4)] = 3

Stack Top = 2

Largest Rectangle = 10

{}

POP

Left Offset = 1
Right Offset = 5
Rectangle =
[2 x (5 – 1)] = 8

Stack Top = 1

Largest Rectangle = 10

{}

POP

Left Offset = 0;
Right Offset = 5
Rectangle =
[1 x (5 – 0)] = 5

Stack Top = EMPTY

Largest Rectangle = 10

{}

SUMMARY

- Monotonic stack is just a stack
- The elements in stack are ordered (increasing or decreasing order)
- Can efficiently find out the closest element on the left or on the right which is greater or less than the present element
- Each element pushed into stack or popped from stack once, the total time complexity is O(n)

PRACTICE PROBLEMS

MWC '15 #2 P2: Towering Towers

COI '06 #1 Patrik

City Game

COCI '14 Contest 6 #5 Neo

VM7WC '15 #2 Gold - Uniting the Earth Empire